EXPRESS MAIL NO. EL403200673US
ATTORNEY DOCKET NO. 17101.0003U1
PATENT

TO ALL WHOM IT MAY CONCERN:

Be it known that we:

Stephen N. Ober, residing at
15 Ledge Hill Road
Southboro, Massachusetts 01772

John Grubmuller, residing at
14 Orchard Hill Circle
Bedford, New Hampshire 03110

Maureen Farrell, residing at
9 Copeland Drive
Bedford, Massachusetts 01730

Charles Wentworth, residing at
45 Birch Street, Unit 3
Attleboro, Massachusetts 02703

Tom Gilbert, residing at
103 Hanlon Road
Holliston, Massachusetts 01746

Kevin Barrett, residing at
32 Cider Hill Lane
Sherborn, Massachusetts 01770

Steven Davis, residing at
2 Auburn Court
Brookline, Massachusetts 02446

Erik Nordman, residing at
23 Lord Street, Apt. 2
Waltham, Massachusetts 02451

Randell Grenier, residing at
11A Wycoma Way
Waltham, Massachusetts 02451

all citizens of the United States of America, have invented certain new and useful improvements in a:

SYSTEM AND METHOD FOR GENERATING DE-IDENTIFIED HEALTH CARE DATA

of which the following is the specification.

Attorney Docket No.: 17101.0003U1

SYSTEM AND METHOD FOR GENERATING DE-IDENTIFIED HEALTH CARE DATA

CROSS-REFERENCE TO RELATED APPLICATION

This application claims the benefit of U.S. Provisional Application Serial No. 60/154,726, filed September 20, 1999, the entirety of which is incorporated herein by this reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention generally relates to computer systems and databases. More particularly, the present invention relates to a system and method for the gathering and analysis of health-care related data, and specifically the gathering and analysis of information regarding the use of pharmaceuticals by individuals. The present invention also relates to techniques for de-identifying the individuals from such pharmaceutical data, in order to maintain privacy.

2. Description of the Related Art

In the medical information field, pharmaceutical claims are processed on large computer systems which receive claims data for patients who have been prescribed one or more medications and have filed claims with insurance companies (or government entities) in order to have the claim paid by the company or entity. The claims data includes very specific details and attributes about the individuals making the claims. For example, attributes can include name, gender, birth date, address, medical diagnosis, specific drug prescribed, and other drugs the patient is using. Consequently, this data is very useful in assisting marketing research relative to usage of a specific drug and identifying various attributes that impact the usage.

The claims data is typically received at a data "clearinghouse" which can be a database for a specific insurance company or a larger database providing the claim processing service for many insurance companies. Moreover, the claims data that are produced by claimants include a significant amount of data, with millions of new claims being entered into the system each month. Several of the claims data clearinghouses have systems handling many terabytes of claims data. Because of the large size of the data being produced and the large amount of attributes, the data is in an inadequate format for efficient search, retrieval and analysis of specific attributes.

Recently, there have been laws passed that prevent the transmission of personal information associated with individuals, within health care claims data. This legislation particularly prohibits the transfer of specific personal data such as names, addresses and social security numbers. Thus, the claims data is no longer allowed to be transmitted from the clearinghouse to others in raw form with the personal data. Without the personal information to segregate the claims data, it becomes much harder to generate valuable research and market data based upon the unique attributes for specific individuals, such as age, gender and geographic distribution.

It is therefore desirous to provide the ability to efficiently gather information from the claims databases to allow research and analysis of the attributes that effect the pharmaceutical industry. Accordingly, the present invention is primarily directed to systems and methods for overcoming the problems discussed above, as well as related limitations of the prior art.

SUMMARY OF THE INVENTION

In one embodiment, the present invention is directed to a system and method for creating a unique alias associated with an individual identified in a health care database, that allows the aggregation of segregated data for marketing research. The system may include a first data store for storing at least one record where each record has a plurality of identification fields, such as name and birth date, which when concatenated uniquely identify an individual, and at least one health care field corresponding to health care data associated with the individual, such as a medication type. The system may also have a second data store and a processor that selects a record of the first data store, selects a subset of the plurality of identification fields within the selected record, concatenates the selected subset of identification fields, and stores the concatenated identification fields in a record in the second data store along with at least one health care field from the selected record of the first data store. The first data store and the second data store can either be located within the same database or in separate databases.

The health care data stored within the first data store may, in one embodiment, correspond to pharmaceutical claims data. The selected subset may correspond to a specific person in the healthcare database, and the person's last name, birthday, and gender are concatenated to form a unique identifier for that record. The processor may analyze longitudinal and historical records of individuals using individual-level linking methodologies based on the concatenated identification fields and the at least one health care field of each record of the second data store. The health care data also can have personal data removed from the various records such that only medically significant information remains, and the identifier allows the medical information to be segregated such that the individual records are still identifiable.

In order to more efficiently process the tremendous amount of data of the health care records, the processor may perform the further steps of selectively gathering the records from the first data store and selectively manipulating the records into a data cube. The records of the first data store are typically in tabular form, and the process of manipulating the records comprises selectively joining and projecting records from the various tabular records in the first data store to ultimately form a data cube comprised of a table of records. The data cube format allows the processor to more easily perform a search of the health care records, and also generate a report by displaying the records of a specific data cube.

The present invention thus provides a method for creating a unique alias associated with an individual identified in a health care database, wherein the health care database stores at least one record, and each record has a plurality of identification fields which when taken together uniquely identify an individual, and at least one health care field may correspond to health care data associated with the individual. The method includes the steps of selecting a record within the health care database, selecting a subset of the plurality of identification fields within the selected record, concatenating the selected subset of identification fields, and storing the concatenated identification fields in a record in a second database with the at least one health care field from the selected record of the first data store. The method preferably includes the step of analyzing longitudinal, historical records of individuals using individual-level linking methodologies based on the concatenated identification fields and the at least one health care field of each record of the second database.

The step of selecting a record within the health care database may comprise selecting a record from pharmaceutical claims data. Further, the step of concatenating the selected subset of identification fields may comprise, for example, concatenating, for a specific person in the healthcare database, that person's last name, birthday, and gender. Thus, based on the concatenated identification fields and the at least one health care field of each record of the second data store, the method may include the step of analyzing longitudinal, historical records of individuals using individual-level linking methodologies.

As discussed above, the method further may include the steps of selectively gathering the records from the first data store, and selectively manipulating the records into a data cube. The step of selecting a record within the health care database may comprise selecting records of the first data store that are in tabular form, and the step of selectively manipulating the records into a data cube may comprise selectively joining and projecting records from the first data store and creating a data cube comprising a table of records.

The data cube allows the present system to aggregate the records in an efficient format such that all new records can be viewed shortly after posting. Further, the unique population identifiers allow users to follow patients over time yielding important results unavailable in other databases, such as patient drug switching behavior. By linking medical and pharmacy transactions at the patient level, new insights such as indication specific use of drugs and patient comorbidities can be determined.

The report displayed by the system may contain several attributes, such as: market shares geographic information at the national, regional, state and MSA levels; trends over time including annual, quarterly, monthly, and weekly periods; traditional measures such as total, new and refilled prescription counts; source of business such as new prescription starts, switches, and continuing patients; prescriber specialty; patient demographics for age and gender; indication specific use; and patient comorbidities. The system can therefore be used in a number of ways to help make business decisions, such as monitoring new drug launches and marketing campaigns, enhanced sales force targeting, and micro-marketing in select geographic areas or to select customers. Furthermore, the system can be used for forecasting and development of a pharmaceutical marketing strategy including indication-specific product positioning, early warning market share shifts, clinical trial site selection, investigator recruiting, and accurate intelligence on market size and demand.

Other objects, features, and advantages of the present invention will become apparent from the drawings, detailed description of the invention , and the claims, below.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGS. 1A, 1B, 1C and 2 are block and flow diagrams showing the overall structure and overall flow of the present invention in one embodiment.

FIG. 3 is a flow and relationship diagram, showing the overall flow and data relationship of the present invention.

FIG. 4 illustrates the operation of the Daily Rx Load Process of the present invention.

FIG. 5 illustrates the operation of the Monthly Mx Load Process of the present invention.

FIG. 6 illustrates the operation of the Monthly Hx Load Process of the present invention.

FIG. 7 illustrates the operation of Quarter Monthly Rx Merge Process of the present invention.

FIG. 8 illustrates the operation of the Prepare Mx Data Process of the present invention.

FIG. 9 illustrates the operation of the Produce Patient Data Process of the present invention.

FIG. 9A illustrates the operation of the RSTRANSFORMER process of the present invention.

FIG. 10 illustrates the operation of the Pull Cube Data Process of the present invention.

FIG. 11 illustrates the operation of the Generate TC Cube Data Process of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

With reference to the drawings, in which like numerals represent like elements throughout, FIGS. 1A, 1B and 2 illustrate a high-level combined block/flow diagram for the present invention. These figures represent both the elements of a block diagram for, as well as the steps performed by the system of, the present invention.

Referring to FIGS. 1A, 1B and 2, the primary processing that takes place in the present invention may be performed by, for example, a high-performance computing system, such as a Sun Microsystems ES10000 computer (at SITE 2). On a periodic basis, such as each day, seven days per week, a computing system at SITE 1 places healthcare claims data at step 103 via a secure connection 190 onto a computer system at SITE 1. This healthcare claims data may include, for example, pharmaceutical, medical, and hospital claims 101 that have been "de-identified" at step 102 (explained in further detail below).

The claims data is de-identified at step 102 before it is sent to SITE 2, which includes applying a unique identifier, encrypting this identifier, and removing specific patient identifying fields. Data is then loaded into database tables (such as an Oracle database) at step 104 that also reside on SITE 2. At step 105, SITE 2 runs all processes for analyzing and consolidating the data and for transforming the resulting Oracle tables into OLAP cubes.

The cube building process may run on a different computer (such as SITE 2). Cubes are modeled using an OLAP product on a desktop computer under, for example, the Windows NT operating system.

The cube deployment process may run on a different computer (such as SITE 3). A computing system at SITE 2 places cubes and metadata files at step 106 via a secure connection to SITE 3. Processes run at step 107 at SITE 3 to place the cube on the production web site and to update the web site pages with the associated metadata.

The present process performed at SITE 2 after obtaining data from the SITE 1 computer, making data ready for cube transformers, and then displaying it on the web at SITE 3 can be logically divided into six major steps, as shown in FIG. 3.

1. Load Oracle Tables (step 301)
2. Produce Patient Data (step 302)
3. Pull Cube Data (step 303)
4. Generate Cube Data (step 304)
5. Build Cube (step 305)
6. Automated Cube Deployment and Metadata Update Process

All these processes are handled, maintained and executed at regular daily, weekly and monthly intervals. There are some processes which are done out of the routine process, such as generation of DOI, zip-state-region, ICD9, etc. tables. FIG. 3 shows a high level overview of the processes used to create cubes.

1. Load Oracle Tables (Step 301)

The Load Oracle Tables process (step 301) can be divided into two logically different steps, daily and monthly processes, described in further detail below with respect to FIGS. 4-8. The daily routines convert the text format data supplied from SITE 1 into "RX" and "Do Not Use Company Name (DNU)" daily Oracle tables. The monthly processes convert Hospital (HX) and Medical (MX) data into monthly Oracle tables. Note that all run times provided below correspond to approximate run times.

1.1 Daily Rx Load Process 401

The Daily Rx Load Process 401 is described below with respect to FIG. 4:

Script Use	Loaddaily.sh is the unix shell script that uses the SQL Loader Rx Control file to convert the Rx Text file from SITE 1 into

	LOAD_YYYYMMDD, {DNU}_YYYY MMDD Oracle tables after doing all the necessary number, char and date conversions. The {DNU} list contains BLUFFCREEK, KROGER, OMNI, PCN, VIP and WALGREENS.
Input	YYYYMMDD.synergy.log.gz,
	RX Control file, YYYYMMDD.ctl.
Output	LOAD_19991123, 24 etc. tables for each day of a month.
	WALGREENS_YYYYMMDD etc. tables for each DNU company.
	../log/YYYYMMDD.log
	../data/YYYYMMDD.synergy.log
	../bad/YYYYMMDD.synergy.bad
	../discard/YYYYMMDD.synergy.discard.
Frequency	Daily
Run Time	~4 hours

1.2 Monthly Mx Load Process 501

The Monthly Mx Load Process 501 is described below with respect to FIG. 5:

Script Use	SQL LOADER process reads MXDaily.MMDDYY text file and control file to convert it into LOAD_MX_YYYYMM tables.
Input	/raid/4011/envoydata/mx/oct1999/data/MXDaily.100199
	MX Control file, YYYYMMDD.ctl.
Output	LOAD_MX_YYYYMM table.
Frequency	Monthly
Run Time	~8 hours

1.3 Load HX Text Data 601

The Load HX Text Data Process 601 is described below with respect to FIG. 6:

Script Use	SQL LOADER process reads HX text file and Control File to convert it into WH_ENVOY_HX_SEP99 tables.
Input	/raid/4011/envoydata/hx/sep1999/data/HCDS.DPRET60.090199, HX Control file, YYYYMMDD.ctl.
Output	WH_ENVOY_HX_SEP99_10..20..30..36..40..46..50.. 60..61..66..70..80..90 tables for HX.
Frequency	Monthly
Run Time	~8 hours

1.4 Quarter-Monthly Rx Merge 701

The Quarter-Monthly Rx Merge Process 701 is described below with respect to FIG. 7:

Script Use	This process uses RX_Weekly_New.sql SQL script to combine all the daily (approx. 8 days of tables) RX and DNU tables into quarter-monthly tables.
Input	LOAD_19991123..24 etc. tables for each day of a month. WALGREENS_YYYYMMDD etc. tables for each "DNU" company.
Output	WH_ENVOY_9911A..B..C..D etc. 4 tables for a month. WALGREENS_9911A..B..C..D like tables for each "DNU" company for a month.
Frequency	Monthly
Run Time	~ 6 hours

1.5 Prepare Mx Data (801)

The Prepare Mx Data Process 801 is described below with respect to FIG. 8:

Script Use	This process uses Process_Monthly_MX.sql SQL script to validate and convert LOAD_MX_YYYYMM table data into required date, char and numbers.
Input	LOAD_MX_YYYYMM, WHREF_DONOT_USE_MX
Output	WH_ENVOY_MX_YYYYMM BAD_PAYER_ID_YYYYMM
Frequency	Monthly
Run Time	

2. Produce Patient Data (step 302)

The Produce Patient Data Process of step 302 (FIG. 3) is described below in further detail with respect to FIG. 9:

Script Use	This process uses Master_PXpgmv1b_9910.sql SQL script to combine weekly RX and monthly MX, HX tables to create a relational WHREF_ENVOY_PXYYMM table.
Input	WH_ENVOY_YYMMA..B etc., WH_ENVOY_MX_YYYYMM, WH_ENVOY_HX_MMMYY_20 tables.
Output	WHREF_PATIENT_REPOSITORY_RXMM, WHREF_MXTEMP_YYMM,

WHREF_HXTEMP_YYMM,

WHREF_ENVOY_PXYYMM tables.

Frequency	Monthly
Run Time	~13 hours

3. Pull Cube Data (step 303)

The Produce Patient Data Process of step 303 (FIG. 3) is described below in further detail with respect to FIG. 10:

This process uses a series of Oracle stored procedures to allow for error checking and audit logging. Logging for these procedures uses the MM_LOG table. These stored procedures are called from the Unix shell using shell script wrappers that input the necessary variable values. The stored procedures used are as follows:

mm00_init

mm01_set_vars

mm02_weekly_data_pull

mm03_memids

mm04_mx_diags

3.1 Audit Logging in Oracle Table MM_LOG

Structure of the MM_LOG table.

RUN_DATE	START_TIME	STOP_TIME	CUBE_NAME	PROCESS	RETURN_CODE	ERROR_CODE	DESCR
10-Jul-00	8:26:37	8:26:38		mm00_init()	0	Completed	Procedure mm00_init() completed successfully.
10-Jul-00	8:26:38	8:26:38		mm01_set_vars()	0	Completed	Procedure mm01_set_vars() completed successfully.
11-Jul-00	8:26:38	12:35:49		mm02_weekly_data_pull()	0	Completed	Procedure mm02_weekly_data_pull() completed successfully.
11-Jul-00	2:04:59	12:11:57		mm03_memids()	0	Completed	Procedure mm03_memids() completed successfully.
11-Jul-00	1:07:32	11:23:46		mm04_mx_diags()	1	-904	ORA-00904: invalid column name

A record is added to MM_LOG for each process. The name of the process is in the PROCESS column. For cube specific processes, the name of the cube is in the CUBE_NAME column. When a process successfully completes, the RETURN_CODE column contains a 0; when there is an error, the RETURN_CODE column contains a 1.

3.2 Initialization

Script Use	The mm00_init procedure initializes the environment for weekly Market Monitory cube processing. The mm00.sh shell script calls the mm00_init procedure.
Input	None
Output	MM_LOG table truncated.
	MM_VARS table truncated.
	CUBE_DATA_TEXT table truncated.
	MM_LOG table – row inserted showing successful completion or error condition.

3.3 Set Variables

Script Use	The mm01_set_vars procedure sets variables for the Rx Market Monitor weekly cube processing. The mm01.sh shell script calls the mm01_set_vars procedure with input variables set as text. The mm00_init procedure must already have been run.	
Input	p_run_date	Run date of pull as text 'YYYYMMDD'.
	p_start_date	Start date of pull as text 'YYYYMMDD'.
	p_end_date	End date of pull as text 'YYYYMMDD'.
	p_post_date	Post date of pull as text 'YYYYMMDD'.

	p_acute_lookback	Acute lookback date as text 'YYYYMMDD'.
	p_chronic_lookback	Chronic lookback date as text 'YYYYMMDD'.
Output	MM_VARS table – row inserted with this week's values as DATE datatype.	
	MM_VARS_HIST table – row inserted with this week's values as DATE datatype.	
	MM_LOG – row inserted showing successful completion or error condition.	

3.4 Pull Weekly Data

Script Use The mm02_weekly_data_pull procedure pulls one week of Rx data for weekly Rx Market Monitor cube processing. The mm02.sh shell script calls this procedure with the tablespace variable input set. The mm00_init and mm01_set_vars procedures must already have been run.

Input	p_tablespace	Tablespace name as text.
	MM_VARS table	
	WH_ENVOY_YYMM where YYMM is the two character year and month from start_date in MM_VARS table.	
	WWW_MASTER_DOI	

Output Last week's WEB_DATA_WEEK_PULL table is renamed to WEB_DATA_WEEK_PULL_YYYYMMDD where YYYYMMDD is one day before the start_date in MM_VARS table.

New WEB_DATA_WEEK_PULL table is created in the WEB schema in the tablespace named in the p_tablespace parameter. The WEB_DATA_WEEK_PULL table contains Rx data from the start and end dates in the MM_VARS table.

MM_LOG – a row is inserted to indicate either successful completion or error condition.

3.5 Get Memids

Script Use The mm03_memids procedure accumulates six weeks of memids. The mm03.sh shell script calls this procedure and inputs the tablespace parameter. The mm00_init, mm01_set_vars, and mm02_weekly_data_pull procedures must already have been run.

Input

p_tablespace	Tablespace name as text.

MM_VARS table

ALL_MEM_TO_CONVERT table

WEB_DATA_WEEK_PULL_V2 table

WEB_UMEMS_WEEK_V2_MONDD table where MONDD is the end_date from MM_VARS table as text.

WEB_UMEMS_WEEK_V2_MONDD[1-5] tables where MONDD[1-5] are the previous five weeks of data.

RXMEMID_SEQ table

Output WEB_UMEMS_WEEK_V2_MONDD table is created where MONDD is start_date in MM_VARS table minus one day.

WEB_UMEMS_WEEK_PULL is created with data for current week and previous 5 weeks.

MM_LOG – a row is inserted to indicate either successful completion or error condition.

3.6 Get Mx Diagnoses

Script Use	The mm04_mx_diags procedure gets diagnoses information from Mx tables for weekly processing. The mm04.sh shell script executes this procedure with the tablespace input variable set. The mm00_init, mm01_set_vars, mm02_weekly_data_pull, and mm03_memids procedures must already have been run.	
Input	p_tablespace	Tablespace name as text.
	MM_VARS table	
	WEB_UMEMS_WEEK_PULL_V2 table	
	MASTER_PX table	
	WEB_MXPTS_WEEK_PULL_V2 table	
	WH_ENVOY_MX_YYYYMM[1-3] tables where YYYYMM[1-3] are the current month and two prior months of data.	
	WEB_RXMX_WEEK_PULL_V2 table	
	RXMEMID_SEQ table	
Output	ACUTE_RXMX table – records from the week are appended.	
	CHRONIC_RXMX table – records from the week are appended.	
	MM_LOG table – row inserted to indicate either successful completion or error condition.	

4. Generate TC Cube Data (step 304)

The Generate TC Cube Data Process of step 304 (FIG. 3) is described below in further detail with respect to FIG. 11:

The Generate TC Cube Data Process 304 uses three Oracle stored procedures to generate a cube table which will be further used by data transformers to build a COGNOS readable multi-dimensional formatted cube structure. The last stored procedure updates statistics for each cube. The stored procedures are as follows:

mm05_step1

mm06_step2

mm07_step3

mm08_cube_metadata

4.1 Process Step 1101 (step 1)

Script Use The mm05_step1 procedure must be run for each therapeutic class. This procedure inserts records into the CMID_V2_CLASS table where CLASS is the p_class specified. The mm00_init, mm01_set_vars, mm02_weekly_data_pull, mm03_memids, and mm04_mx_diags procedures must already have been run.

Input	p_class	Class name.
	p_tablespace	Tablespace name as text.
	p_lookback	Number of days of lookback
	p_condition	"ACUTE" or "CHRONIC"
	MM_VARS table	
	WEB_DATA_WEEK_PULL_V2 table	
	CUBE_V2_LIST table	
	CMID_V2_CLASS table where CLASS is the p_class.	

Output Records inserted into CMID_V2_CLASS table where CLASS is the p_class.

New CMID_V2_CLASS_TMP table is created where CLASS is the p_class.

MM_LOG table – row inserted to indicate either successful completion or error condition.

4.2 Process Step 1102 (step 2)

Script Use The mm06_step2 procedure must be run for each therapeutic class. This procedure inserts records into the RX_RESULT_V2_CLASS table where CLASS is the p_class specified when the procedure is called. The mm00_init, mm01_set_vars, mm02_weekly_data_pull, mm03_memids, mm04_mx_diags, and mm05_step1 procedures must already have been run.

Input

p_class	Class name.
p_tablespace	Tablespace name as text.
p_lookback	Number of days of lookback
p_condition	"ACUTE" or "CHRONIC"

MM_VARS table

CMID_V2_CLASS_TMP table where CLASS is the p_class.

CMID_V2_CLASS table where CLASS is the p_class.

RX_RESULT_TEMP_CLASS table where CLASS is the p_class.

ZIP_ST_MSA_REG_DIV table

WEB_DEA_TO_SPEC_U

Output New records are inserted into RX_RESULT_V2_CLASS where CLASS is p_class.

MM_LOG table – row inserted to indicate either successful completion or error condition.

4.3 Process Step 1103 (step 3)

Script Use	The mm07_step3 procedure must be run for each therapeutic class. This procedure creates a new RXMX_CUBE_V2_CLASS table where class is the p_class specified. The mm00_init, mm01_set_vars, mm02_weekly_data_pull, mm03_memids, mm04_mx_diags, mm05_step1, and mm06_step2 procedures must already have been run.	
Input	p_class	Class name.
	p_tablespace	Tablespace name as text.
	p_lookback	Number of days of lookback
	p_condition	"ACUTE" or "CHRONIC"
	RXMX_CONDITION_V2 table where CONDITION is the p_condition.	
	RX_RESULT_V2_CLASS table where CLASS is the p_class.	
	ICD9_V2_CLASS table where CLASS is the p_class.	
	RX_RESULT_V2_CLASS_M table where CLASS is the p_class.	
Output	New RXMX_CUBE_V2_CLASS table is created where CLASS is p_class.	
	MM_LOG table – row inserted to indicate either successful completion or error condition.	

4.4 Generate Cube Metadata

Script Use The mm08_cube_metadata procedure must be run for each therapeutic class. This procedure updates the CUBE_DATA table for each cube. The mm00_init, mm01_set_vars, mm02_weekly_data_pull, mm03_memids, mm04_mx_diags, mm05_step1, mm06_step2 and

	mm07_step3 procedures must already have been run.	
Input	p_class	Class name.
	MM_VARS table	
	CUBE_DATA table where CLASS is the p_class.	
Output	CUBE_DATA_TEXT table is appended where CLASS is p_class.	
	MM_LOG table – row inserted to indicate either successful completion or error condition.	

5. Build Cube (step 305)

The Build Cube Process of step 305 (FIG. 3) is described below in further detail with respect to FIG. 9A:

This process uses a C program to create a cube for each therapeutic class. Each cube is FTP'd to the server of SITE 3. Metadata for each cube is spooled to a text file and FTP'd to the SITE 3 server. The same text files may be concatenated and sent via email to the web developer of SITE 2.

5.1 Build Cube

Script Use	The program RSSERVER is repeated for each of the therapeutic classes and called by the script mmv2_'class name'.sh. Data Transformers uses Model Structure and OBES_RX_CUBE Oracle table (built in above process) to finally build a CUBE for each of the therapeutic class. This Cube is then used by COGNOS to show requested information on the Web.
Input	OBES_RXMX_CUBE
Output	Cube for a OBES Class
Frequency	On Request

Run Time ~8 hours

5.2 FTP Cube to SITE 3 server

Script Use The transfer_mm_cube.sh script renames a cube and puts a copy into directory **/raid4011/cubes/transfer** where it will automatically be FTP'd to the SITE 3 server. This script is run in parallel for each class (class corresponds to cube).

5.3 Approve Cube

Script Use The approve_cube script is run manually for each cube after quality assurance has been performed. This script is run in parallel for each class (class responds to cube).

5.4 Create metadata text file/FTP to SITE 3 server

Script Use The process gen_mm_file.sql is called by gen_mm_file.sh to spool metadata for a cube to a text file. The text file is put into a directory where it is automatically FTP'd to the SITE 3 server. This script is run in parallel for each class. All procedures to pull cube data must have been successfully completed and the metadata must exist in the Oracle tables cube_data and cube_data_text.

5.5 E-mail Metadata to Web Developer

Script Use The email_meta.sh script will e-mail metadata to a SITE 2 Web

Developer. This script is run in parallel for each class (class corresponds to cube).

6. Automated Cube Deployment and MetaData Update Process

Automated processes exist on the OnLine Analytical Processing (OLAP) host machine to deploy data cubes (such as QUINTERNET™ Series, from Quintiles Transnational Corp.) to the production web site, cubes ready for Quality Assurance (QA) verification, as well as to automatically update "metadata" on production web pages. This enables production cube deployments and web page updates to occur during off-peak hours without any manual intervention.

As a QUINTERNET™ Series data cube is created, the cube is sent via a secure connection to the host machine. The cube is then automatically "served up" to the QA location on the web, to which only authorized personnel have access.

For each cube approval, a "metadata" file is transmitted from the SITE 2 server, via a secure connection, to the host machine in a specific location (directory within a file system). This secure transmission may occur after a data cube has passed the QA verification.

The metadata file contains statistical information about the specific cube (e.g.- date that cube contains data through, number of records, number of patients, etc.). Several times each night, an automated process may be initiated which checks for the presence of a metadata file and a corresponding data cube file. If matching files for a specific cube exist, the process automatically "serves" up this cube into production for access via the web pages. In addition, the HTML page which contains the metadata for the cube is updated with the metadata contained in the metadata file.

The server at, for example, SITE 3 may prepare and maintain HTML template files for each QUINTERNET™ Series cube. These files contain the base HTML used to create each cube's web page. Instead of the actual metadata values that will populate the cubes' web pages, the HTML template files may contain placeholder tags. These placeholder tags are replaced by data values supplied by SITE 2 in metadata files.

SITE 2 transfers the template files and the metadata files to a host via FTP. The metadata files are transferred to the host each time a cube is approved. Template files are maintained for each QUINTERNET™ Series cube and are updated by SITE 2 as necessary so that a current version of each cube's template file is always available for processing on the host.

After a cube has been updated, reviewed for quality and approved by the operator of SITE 2, SITE 2 transfers a metadata file for that cube to the host via FTP. The metadata files contains the same tags found in the HTML template file for each cube. Each of these tags is coupled with a value that will be substituted for the placeholder tag in the HTML template file.

An event-driven file processing script runs periodically via cron, a unix scheduling system, on the host. If the file processing script detects the existence of a designated flag file, a script called enable_cube.ksh is run. The enable_cube.ksh script calls a Perl script, replaceHtmlMetaTags.pl, passing it the name of the cube being processed and the name of the related metadata file. The enable_cube.ksh script also updates the metadata file with a tag/value pair representing the date the updated cube is being deployed.

The purpose of the replaceHtmlMetaTags.pl script is to automatically generate HTML pages for the QUINTERNET™ Series products. The replaceHtmlMetaTags.pl script substitutes the values in the metadata file for the placeholder tags in the template and saves the resulting output in an HTML file. Referring to FIG 1C, the enable_cube.ksh script then promotes the updated HTML file(s) to SITE 3's web server 181 thus making it available via, for example, the Internet 183, to users of web browsers 182 operating on client computers.

The present invention may be implemented with a processing schedule defined in many ways. For example, the schedule may be on a weekly or monthly basis, depending upon the needs of the implementation. At times, special requests may be required and the ability to process data and create cubes on an ad hoc basis exists.

While there has been shown the preferred embodiment of the present invention, it is to be understood that certain changes can be made in the forms and arrangements of the elements of the system and the steps of the method without departing from the spirit and scope of the invention as is set forth in the Claims.